Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A random number generator comprising:

a plurality of cross-connected latches providing at least two latch outputs;

~~at least one input of one latch of the plurality of latches being driven by a clock signal;~~

~~a first XOR~~ first XOR logic that receives input signals derived from the at least two latch ~~outputs as an input;~~ outputs, wherein said first XOR logic generates a current value of a mistake signal "E" when ~~its inputs~~ the input signals do not match ~~from the at least two latch outputs being at different logic states~~; and

~~wherein~~ second XOR logic to compare the current value of the mistake signal ~~is compared~~ with a previously stored value of the mistake signal ~~by a second XOR~~ to determine whether to obtain a random bit from a pseudo random stream of bits.

2. (currently amended) The random number generator according to claim 1, further comprising:

an ~~exclusive or~~ XOR network comprising a plurality of strings of cascaded flip-flops, wherein an output of each of the plurality of strings is connected to the first XOR ~~circuit,~~ logic, and wherein each respective latch output of the at least two latch outputs is connected to an input of a respective string of the plurality of cascaded flip-flops.

3. (currently amended) The random number generator according to claim 2, wherein the plurality of strings of cascaded ~~flip-flops~~ flip-flops comprise D flip-flops.

4. (currently amended) The random number generator according to claim 2, further comprising a flip-flop arrangement driven by ~~a third XOR~~ third XOR logic to provide an acquisition signal "A" that is input to the XOR network ~~by providing as~~ a clock input to the strings of cascaded flip-flops.

5. (currently amended) The random generator according to claim 1, ~~wherein~~ further comprising a flip-flop to store a logical value of said previously stored value of the mistake signal ~~mistake is stored in a flip-flop~~.

6. (currently amended) The random generator according to claim 1, ~~wherein~~ further comprising a shift register to store a bit if the previously stored ~~mistake~~ value of the mistake signal disagrees with the current value of the mistake signal ~~mistake "E" then a bit is stored in a shift register~~.

7. (currently amended) The random generator according to claim 5, ~~wherein~~ further comprising a shift register to store a bit if the previously stored ~~mistake~~ value of the mistake signal disagrees with the current value of the mistake signal ~~mistake "E" then a bit is stored in a shift register~~.

8. (currently amended) The random number generator according to claim 7, wherein the bit is stored in the shift register when the previously stored value of the mistake signal is a logical zero.

9. (currently amended) The random number generator according to claim 7, ~~wherein~~ further comprising an AND gate to enable the shift register ~~is enabled~~ for shifting ~~via an AND gate~~, wherein said AND gate ~~has~~ comprises a first input from said second XOR logic and a second input from said flip-flop.

10. (currently amended) The random number generator according to claim 1, ~~wherein the random bit obtained from the pseudo-random stream of bits is generated by~~ further comprising a Linear Feedback Shift Register (LFSR) to generate the pseudo-random stream of bits.

11. (original) The random number generator according to claim 10, wherein the LFSR has at least 64 bits.

12. (original) The random number generator according to claim 7, ~~wherein the random bit obtained from the pseudo-random stream of bits is generated by~~ further comprising a Linear Feedback Shift Register (LFSR) to generate the pseudo-random stream of bits.

13. (original) The random number generator according to claim 12, wherein the LFSR has at least 64 bits.

14. (currently amended) A method of random number generation comprising ~~the steps of~~:

~~(a) providing~~ generating at least two latch outputs from a plurality of cross-connected latches ~~having at least two latch outputs~~;

~~(b) driving at least one input of one latch of the plurality of latches by a clock signal;~~

~~(c) connecting a first XOR that receives~~ receiving input signals derived from the at least two latch outputs as an input to first XOR logic;

~~(i) wherein said first XOR generates~~ generating a current value of a mistake signal "E" when ~~its inputs~~ the input signals do not match ~~from the at least two latch outputs(latch0, latch1) being at different logic states~~; and

~~(ii)~~ comparing the current value of the mistake signal with a previously stored value of the mistake signal ~~by a second XOR~~ at second XOR logic to determine whether to obtain a random bit from a pseudo random stream of bits.

15. (currently amended) The method according to claim 14, ~~wherein step (c) further comprises: (iii)~~ further comprising obtaining the random bit from the pseudo-random stream of bits.

16. (currently amended) The method according to claim 14, further comprising:

providing an ~~exclusive or~~ XOR network comprising a plurality of strings of cascaded flip-flops, wherein an output of each of the plurality of strings is connected to the first XOR ~~circuit,~~ logic, and wherein each respective latch output of the at least two latch outputs is connected to an input of a respective string of the plurality of cascaded flip-flops.

17. (currently amended) The method according to claim 16, wherein the plurality of strings of cascaded ~~flip-flops~~ flip-flops comprise D flip-flops.

18. (currently amended) The method to claim 16, further ~~comprising~~ comprising:

providing ~~a flip-flop arrangement driven by a third XOR to provide~~ an acquisition signal "A" from a flip-flop arrangement driven by third XOR logic; and

~~that is input~~ inputting the acquisition signal "A" to the XOR network ~~by providing~~ as a clock input to the strings of cascaded flip-flops.

19. (currently amended) The method according to claim 14, ~~wherein~~ further comprising storing a logical value of said previously stored ~~mistake is stored~~ value of the mistake signal in a flip-flop.

20. (currently amended) The method according to claim 14, ~~wherein~~ further comprising storing a bit in a shift register if the previously stored ~~mistake~~ value of the mistake signal disagrees with the ~~mistake "E" then a bit is stored in a shift register.~~ current value of the mistake signal "E."

21. (currently amended) The method according to claim 19, ~~wherein~~ further comprising storing a bit in a shift register if the previously stored ~~mistake~~ value of the mistake signal disagrees with the ~~mistake "E" then a bit is stored in a shift register.~~ current value of the mistake signal "E."

22. (currently amended) The method according to claim 21, ~~wherein~~ further comprising storing the bit ~~is stored~~ in the shift register only when the previously stored value of the mistake signal is a logical zero.

23. (currently amended) The method according to claim 21, ~~wherein~~ further comprising enabling the shift register ~~is enabled~~ for shifting via an AND gate, wherein said AND gate ~~has~~ receives a first input from said second XOR logic and a second input from said flip-flop.